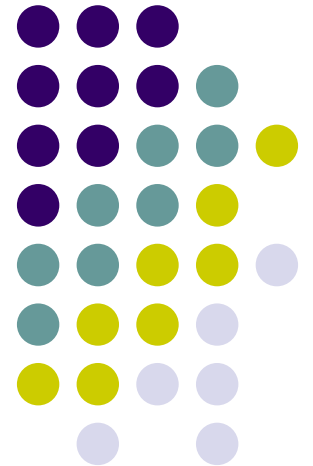
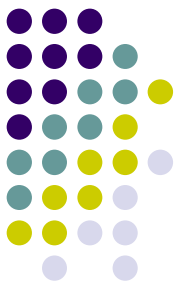


Dimension Reduction

Zhang Hongxin
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2007-03-15





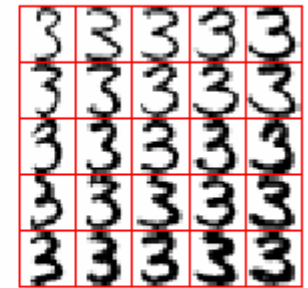
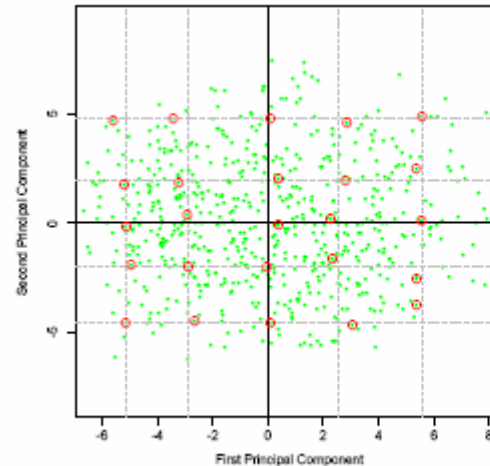
Introduction

- Goal: choosing suitable transforms, so as to obtain high “information packing”.
 - Raw data -> Meaningful features.
 - Unsupervised/Automatic methods.
- To exploit and remove information redundancies via transform.

Feature extraction

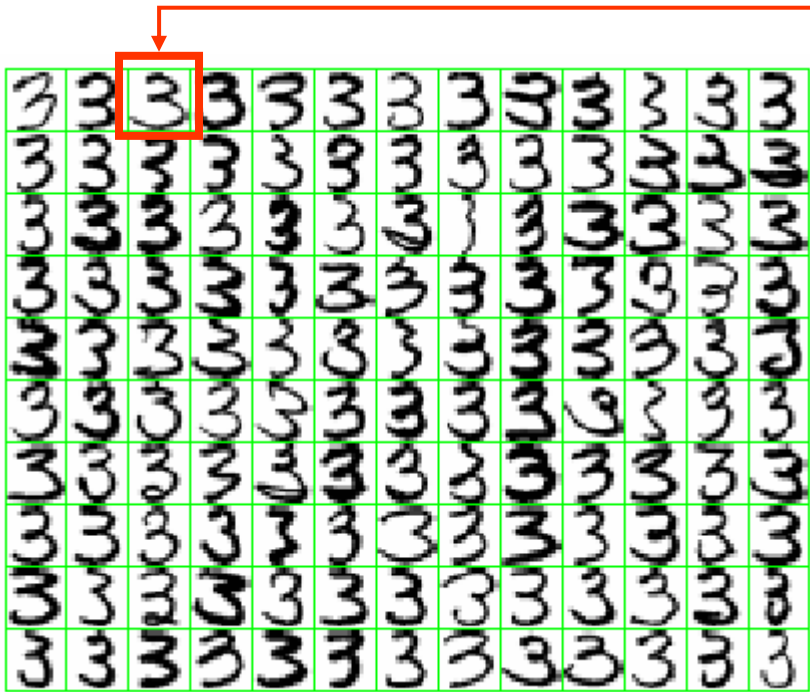


- Data independent
 - DFT, DWT, DCT
 - A single piece of signal
- Data dependent
 - PCA, K-PCA, ICA, ISO-MAP, LLE ...
 - A set of signals (images, motion data, shapes,...)
- Key: define desirable transforms
 - Raw data -> Feature space



PCA: example

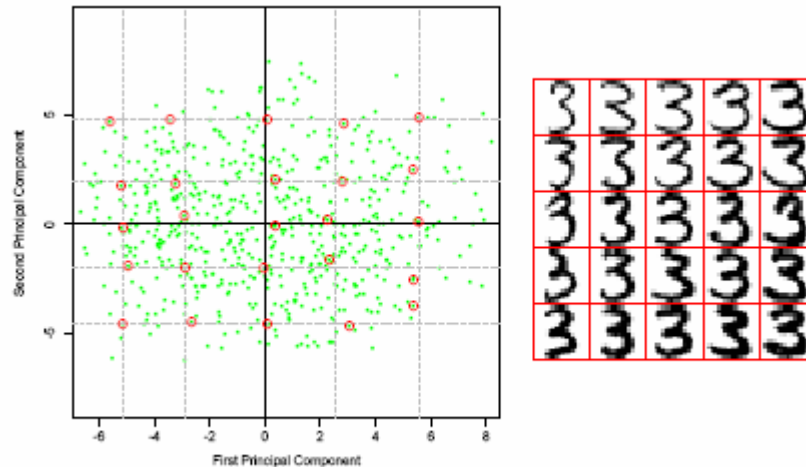
Digit data



$$X = \begin{pmatrix} x_{0,0} & x_{1,0} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & x_{1,1} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0,p-1} & x_{1,p-1} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{N \times p}$$

130 threes, a subset of 638 such threes and part of the handwritten digit dataset. Each three is a 16×16 greyscale image, and the variables X_j , $j = 1, \dots, 256$ are the greyscale values for each pixel.

Digit: rank-2 model for threes



Two-component model has the form

$$\begin{aligned}\hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \boxed{\text{3}} + \lambda_1 \cdot \boxed{\text{3}} + \lambda_2 \cdot \boxed{\text{3}}.\end{aligned}$$

Here we have displayed the first two principal component directions, v_1 and v_2 , as images.



Principal Components

- Suppose we have N measurements on each of p variables X_j , $j=1,2,\dots,p$. There are several equivalent approaches to principal components:
 - Produce a derived (and small) set of uncorrelated variables $Z_k = a_k^T X$, $k=1,\dots,q < p$ that are linear combinations of the original variables, and that explain most of the variation in the original set.
 - Approximate the original set of N points in $\mathfrak{R}^p$ by a least-squares optimal linear manifold of co-dimension $q < p$.
 - Approximate the $q < p$ data matrix X by the best rank- q matrix $\hat{X}(p)$. This is the usual motivation for the SVD.

数据的简化
线性表示

数据的低维线
性流形近似

矩阵
逼近



Basis Vectors and Images

- Input samples

$$\mathbf{X}^T = [X(1), X(2), \dots, X(p)]$$

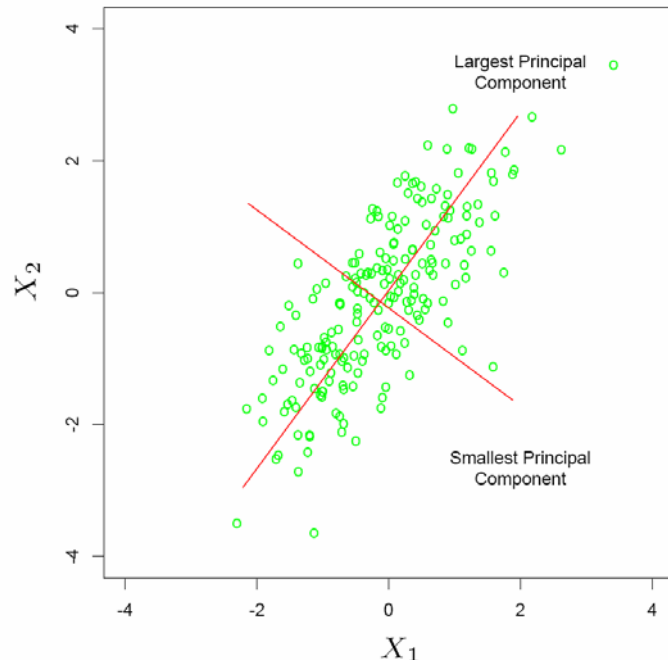
- Unitary $p \times p$ matrix A and transformed Vector
 $\mathbf{Z} = \mathbf{A}\mathbf{X}$

- Basis vector representation

$$\mathbf{x} = \mathbf{A}\mathbf{z} = \sum_{i=0}^{N-1} z(i)\mathbf{a}_i$$

$$\langle \mathbf{a}_j, \mathbf{x} \rangle = \mathbf{a}_j^T \mathbf{x} = \sum_{i=1}^p z(i) \langle \mathbf{a}_j, \mathbf{a}_i \rangle = z(j)$$

PCA: Derived Variables



$$\Sigma = \mathbf{X}^T \mathbf{X}$$

- $\mathbf{Z}_1 = a_1 \mathbf{X}$ is the projection of the data onto the longest direction, and has the largest variance amongst all such normalized projections.
- a_1 is the largest eigenvalue of Σ , the sample covariance matrix of $\mathbf{X}$. $\mathbf{Z}_2$ and a_2 correspond to the second-largest eigenvector.

PCA: Least Squares Approximation



Find the linear manifold

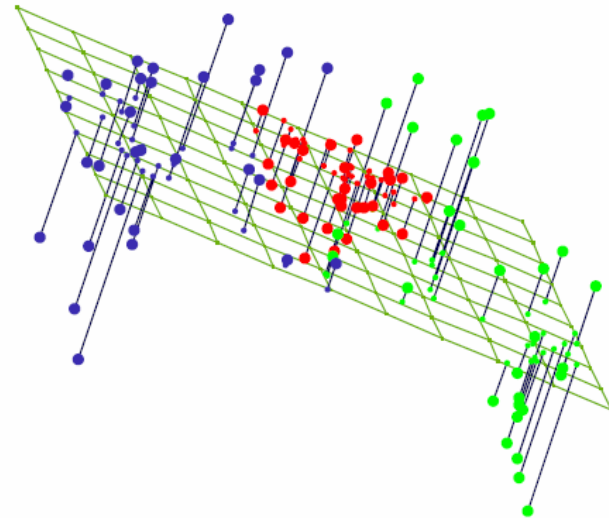
$$f(\lambda) = \mu + V_q \lambda$$

that best approximates the
data in a least-squares
sense:

$$\min_{\mu, \{\lambda_i\}, V_q} \sum_{i=1}^N \left\| \mathbf{x}_i - \mu - V_q \lambda_i \right\|^2$$

Solution:

$$\mu = \bar{\mathbf{x}}, v_k = a_k, \lambda_k = Z_k$$



PCA:

Singular Value Decomposition



Let $\hat{X}$ be the **centered** $N \times p$ data matrix (assume $N > p$).

$$X = \begin{pmatrix} x_{0,0} & \boxed{x_{1,0}} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & \boxed{x_{1,1}} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{0,p-1} & \boxed{x_{1,p-1}} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{N \times p}$$

Singular values

$= USV$

Unitary Matrices

is the SVD of $\hat{X}$, where

x_1

- **U is $N \times p$ orthogonal, the left singular vectors.**
- **V is $p \times p$ orthogonal, the right singular vectors.**
- **S is diagonal, with $d_1 \geq d_2 \geq \dots \geq d_p \geq 0$, the singular values.**
- ✓ **The SVD always exists, and is unique up to signs. The columns of V are the principal components, and $Z_j = U_j d_j$.**

PCA:

Singular Value Decomposition



$$\begin{array}{c}
 \begin{array}{c} \text{Singular values} \\ \downarrow \\ = USV \\ \uparrow \uparrow \\ \text{Unitary Matrices} \end{array} \\
 X = \begin{pmatrix} x_{0,0} & \boxed{x_{1,0}} & x_{2,0} & \cdots & x_{N-1,0} \\ x_{0,1} & \boxed{x_{1,1}} & x_{2,1} & \cdots & x_{N-1,1} \\ \vdots & \boxed{\vdots} & \vdots & \ddots & \vdots \\ x_{0,p-1} & \boxed{x_{1,p-1}} & x_{2,p-1} & \cdots & x_{N-1,p-1} \end{pmatrix}_{N \times p} \\
 \begin{array}{c} x_1 \end{array}
 \end{array}$$

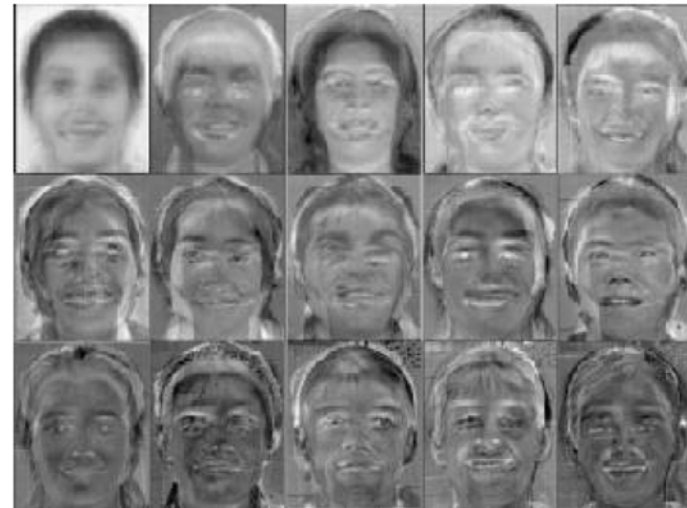
Let s_q be s with all but the first q diagonal elements set to zero. Then $\hat{X}_q = US_qV^T$ solves

$$\min_{\text{rank}(\hat{X}_q)=q} \|\hat{X} - \hat{X}_q\|$$

PCA: example Eigenfaces

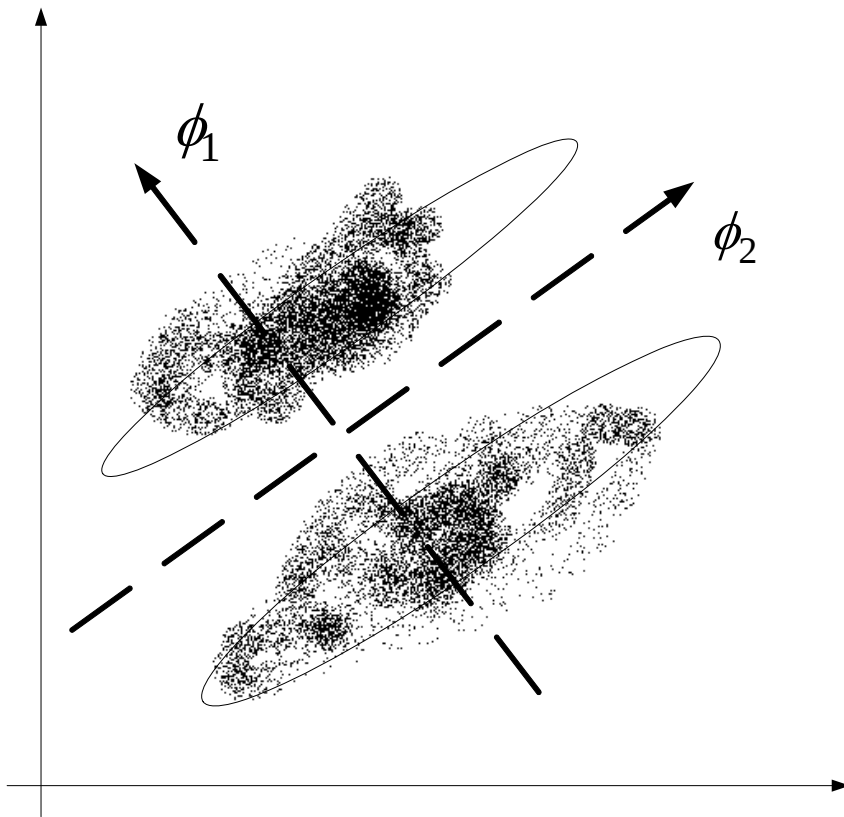
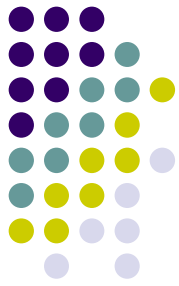


- G. D. Finlayson, B. Schiele & J. Crowley. Comprehensive colour image normalization. ECCV 98 pp. 475~490.



- Eigen-X, ☺

Problems of PCA



- Only suitable for normal distributed data
- More consideration
 - ICA: Independent components.
 - K-PCA: Nonlinear
 - ...

Nonlinear dimension reduction algorithms:



- Locally Linear Embedding (LLE), *Science*
Sam T. Roweis and Lawrence K. Saul
- A Global Geometric Framework for Nonlinear Dimensionality Reduction (Isomap), *Science*
Joshua B. Tenenbaum, Vin de Silva, John C. Langford
- BoostMap: A Method for Efficient Approximate Similarity Rankings, *CVPR 2004*
Vassilis Athitsos, Jonathan Alon, Stan Sclaroff, and George Kollios

Locally Linear Embedding (LLE)



- Recovers global nonlinear structure from locally linear fits.
- Each data point and its neighbors is expected to lie on or close to a locally linear patch.
- Each data point is constructed by its neighbors:

$$\hat{\vec{X}}_i = \sum_j W_{ij} \vec{X}_j$$

$$W_{ij} = 0 \quad \text{if} \quad \vec{X}_j \text{ is not a neighbor of } \vec{X}_i$$



LLE:

Getting the Reconstruction Weights

- We want to minimize the error function:

$$\varepsilon(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- With the constraints:

$$W_{ij} = 0 \quad \text{if } \vec{X}_j \text{ is not a neighbor of } \vec{X}_i$$

$$\sum_j W_{ij} = 1$$

- Solution (using Lagrange multipliers):

$$W_j = \sum_k C_{jk}^{-1} (\vec{X} \vec{\eta}_k + \lambda)$$

$$\lambda = 1 - \sum_{jk} C_{jk}^{-1} (\vec{X} \vec{\eta}_k) / \sum_{jk} C_{jk}^{-1}$$

LLE:

Find Embedded Coordinates



- Choose d-dimensional coordinates, Y , to minimize:
$$\phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j w_{ij} \vec{Y}_j \right|^2$$

Under: $\sum_i \vec{Y}_i = \vec{0}$, $\frac{1}{N} \sum_i \vec{Y}_i \vec{Y}_i^T = I$

Quadratic form:
$$\phi(Y) = \sum_{ij} M_{ij} (\vec{Y}_i \vec{Y}_j)$$

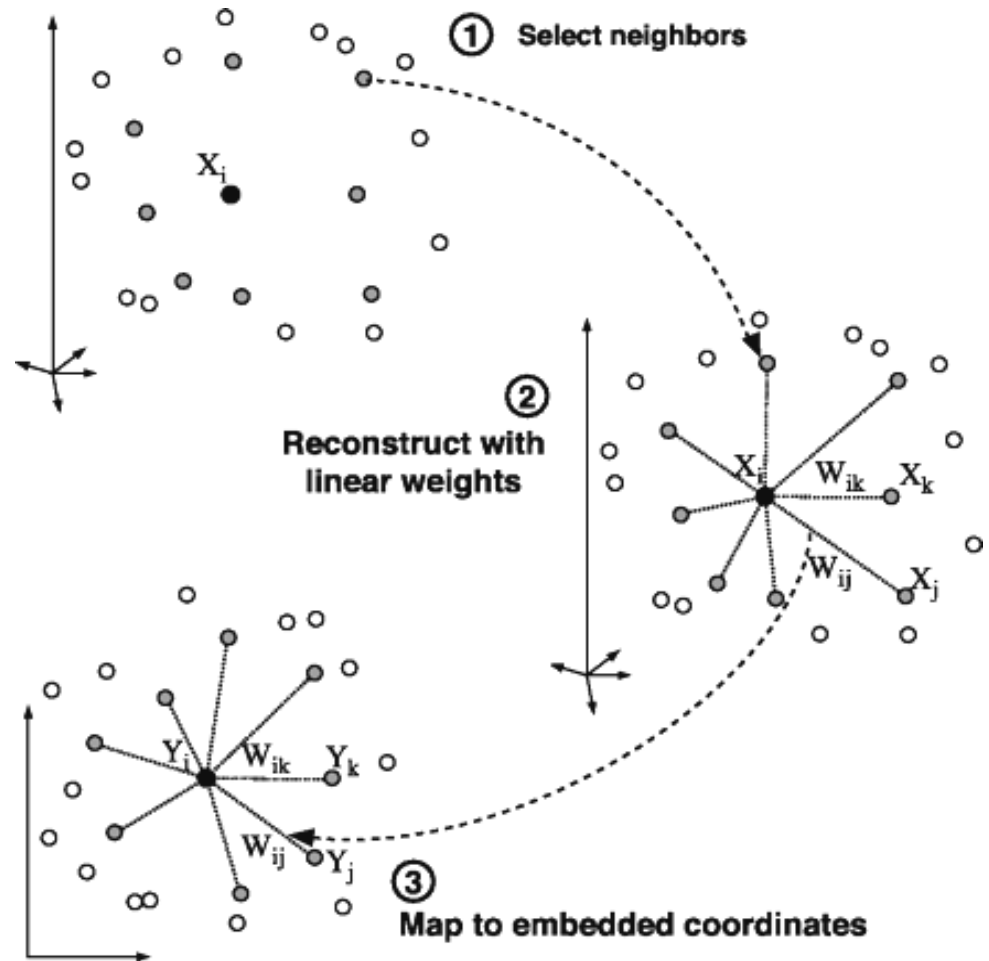
where:
$$M = (I - W)^T (I - W)$$

- Solution: compute bottom d+1 eigenvectors of M . (discard the last one)

LLE: Summary



- Input: N data items in D dimension (X).
- Output: $d < D$ dimensional embedding coordinates (Y) for the input points.



LLE:

Algorithm Pseudocode (I)



Find neighbors in X space

For $i=1:N$

- compute the distance from X_i to every other point X_j

- find the K smallest distances

- assign the corresponding points to be neighbors of X_i

end

LLE:

Algorithm Pseudocode (II)



Solve for reconstruction weights W .

for $i=1:N$

 create matrix Z consisting of all neighbors of X_i

 subtract X_i from every column of Z

 compute the local covariance $C=Z'^*Z$

 solve linear system $C*w = 1$ for w

 set $W_{ij}=0$ if j is not a neighbor of i

 set the remaining elements in the i th row of W equal to $w/\text{sum}(w)$;

end

LLE:

Algorithm Pseudocode (III)



Compute embedding coordinates Y using weights W .

create sparse matrix $M = (I-W)'(I-W)$

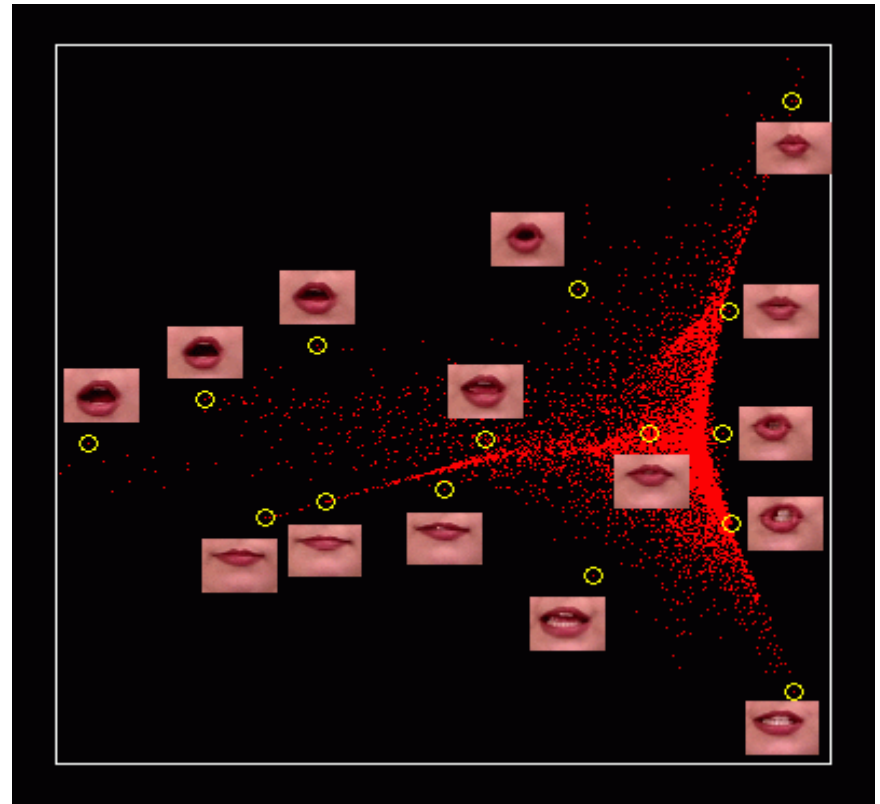
find bottom $d+1$ eigenvectors of M (corresponding to the $d+1$ smallest eigenvalues)

set the q -th ROW of Y to be the $q+1$ smallest eigenvector (discard the bottom eigenvector $[1,1,1,1\dots]$ with eigenvalue zero)

LLE: Example



- $N=8588$ (RGB) images of lips of size 108×84 .
 $D=27216$
- Num of neighbors $K=16$



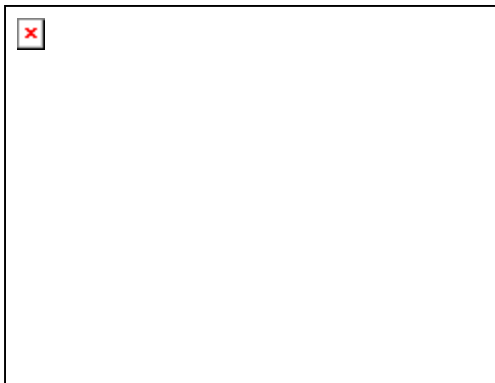
Isomap: (Science 2001)

Isometric feature mapping



- Preserve the intrinsic geometry of the data.
- Use the geodesic manifold distances between all pairs.

Three steps algorithm



Isomap:

Construct Neighborhood Graph



- Determine which points are neighbors, based on the distances $d(i,j)$.
 - K nearest neighbors
 - ϵ -radius
- Create a graph G , with edges between neighbors and distance weights.



Isomap: Compute Shortest Paths



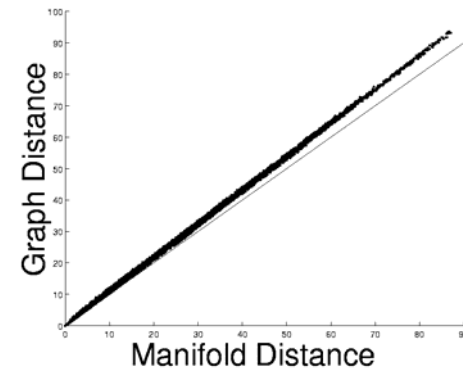
- Estimate the geodesic distances.
- Compute all-pairs shortest paths in G .
- Can be done using Floyd's algorithm, $O(N^2 \ln N)$.

$$d_G(i, j) = d(i, j) \text{ neighboring } i, j$$

$$d_G(i, j) = \infty \quad \text{otherwise}$$

for $k = 1, 2, \dots, N$

$$d_G(i, j) = \min\{d_G(i, j), d_G(i, k) + d_G(k, j)\}$$



Isomap: Construct d-dimensional Embedding



Classical **MDS** with $d_G(i,j)$,
minimize the cost function:

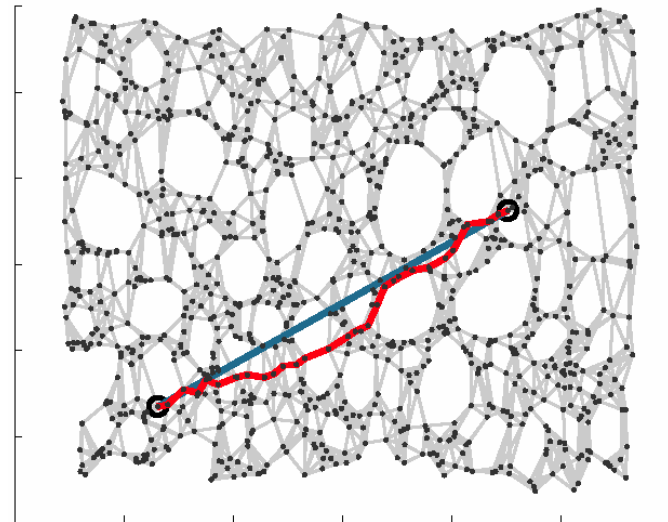
$$E = \left\| \tau(D_G) - \tau(D_Y) \right\|_{L^2}$$

where $D_Y(i, j) = \|y_i - y_j\|$

$$D_G(i, j) = d_G(i, j)$$

and

$$\tau(D) = \frac{-1}{2} \left(I - \frac{1}{N} \right) D^{\cdot 2} \left(I - \frac{1}{N} \right)$$



Solution: take top d
eigenvectors of the
matrix $\tau(D_G)$

Isomap: Classical Multi-dimensional Scaling



$$\mathbf{X}'\mathbf{X} = -\frac{1}{2}\mathbf{J}\mathbf{E}\mathbf{J} \quad \text{E: Euclidian distance matrix}$$

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{M}\mathbf{J} \quad \text{M: Manifold distance matrix}$$

$$\begin{aligned} L(\hat{\mathbf{X}}) &= \left\| -\frac{1}{2}\mathbf{J}(\mathbf{E} - \mathbf{M})\mathbf{J} \right\| \\ &= \left\| \hat{\mathbf{X}}\hat{\mathbf{X}}' - \mathbf{B} \right\|. \end{aligned}$$

$$\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}' \quad \hat{\mathbf{X}} = \mathbf{Q}_+ \mathbf{\Lambda}_+^{\frac{1}{2}}$$

Eigen-structure analysis, SVD again

$$c_i = \sum_{a=1}^m x_{ia}^2$$

$$d_{ij}^2 = \sum_{a=1}^m (x_{ia} - x_{ja})^2$$

$$\mathbf{E} = \mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2\mathbf{X}\mathbf{X}'$$

$$\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'$$

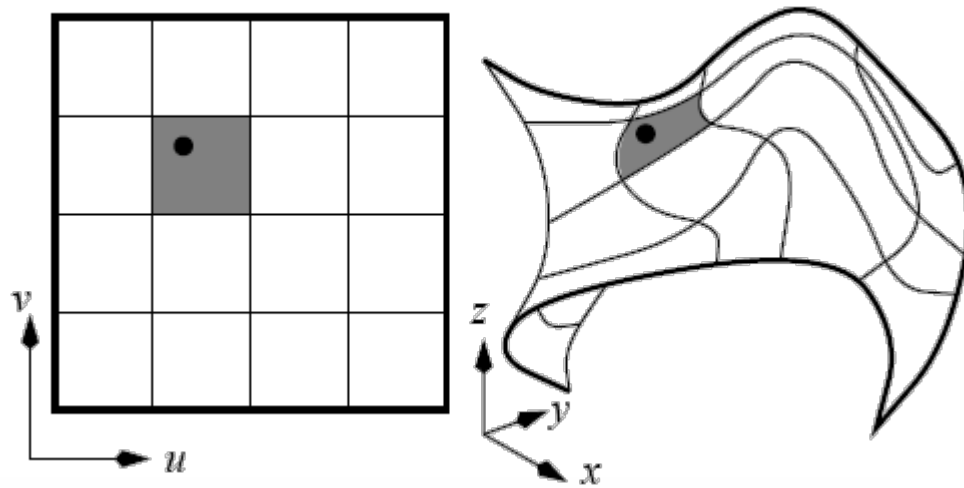
$$\begin{aligned} \mathbf{B} &= -\frac{1}{2}\mathbf{J}(\mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2\mathbf{X}\mathbf{X}')\mathbf{J} \\ &= -\frac{1}{2}\mathbf{J}\mathbf{c}\mathbf{0}' - \frac{1}{2}\mathbf{0}\mathbf{c}'\mathbf{J} + \mathbf{J}\mathbf{X}\mathbf{X}'\mathbf{J} \\ &= \mathbf{X}\mathbf{X}'. \end{aligned}$$

Isomap: Classical Multi-dimensional Scaling (2D)



```
J    = eye(n) - ones(n)./n;  
B    = -0.5 * J * M * J;  
      % Find largest eigenvalues+their eigenvectors:  
[Q, L] = eigs(B, 2, 'LM');  
      % Extract the coordinates:  
newy   = sqrt(L(1, 1)). * Q(:, 1);  
newx   = sqrt(L(2, 2)). * Q(:, 2);
```

Isomap: application texture mapping



(a)



(b)



(a)



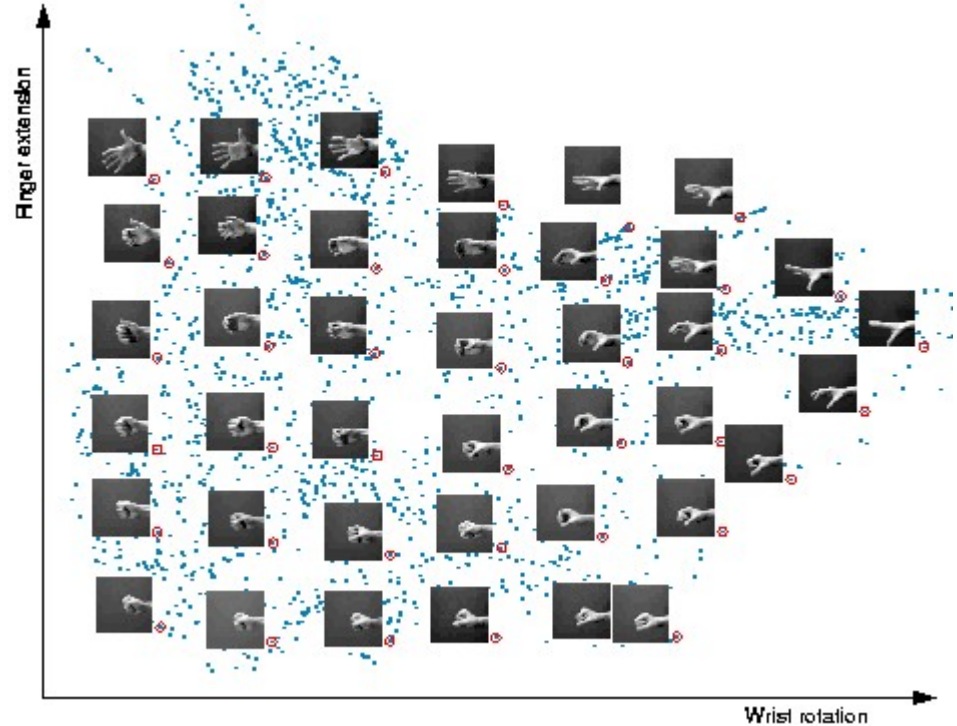
(b)

Fig. 3. An example of a face flattening. (a) A 3D reconstruction of a face. (b) The flattened texture image of the face.

Isomap: Examples



- $N=2000$ images
64x64 pixels $K=6$



Isomap: More Results

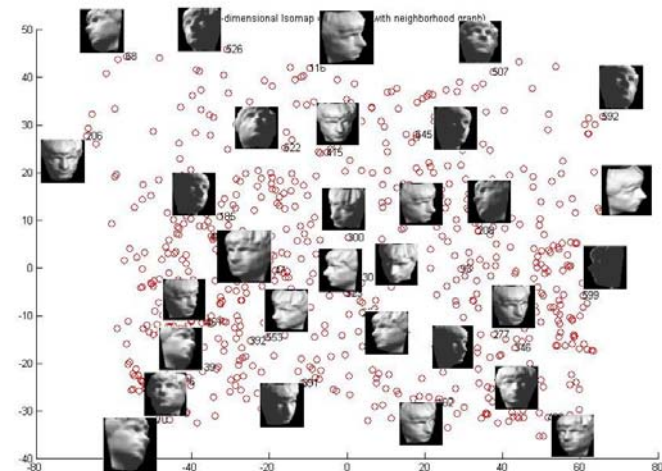
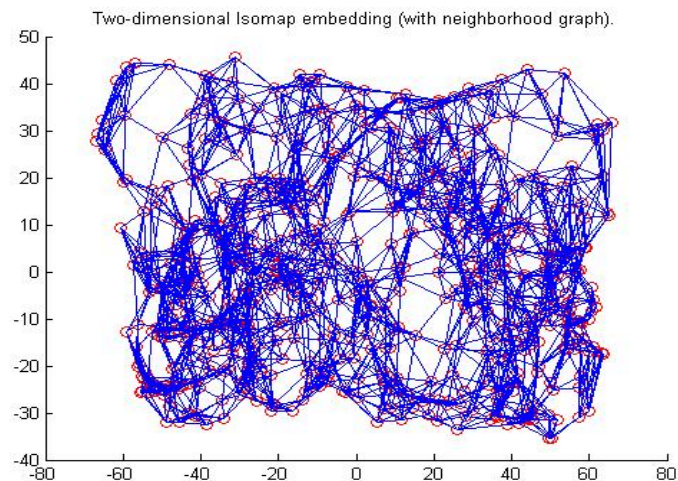


Input: 698
images of 64x64

$K=7$, $d=2$



Outputs:

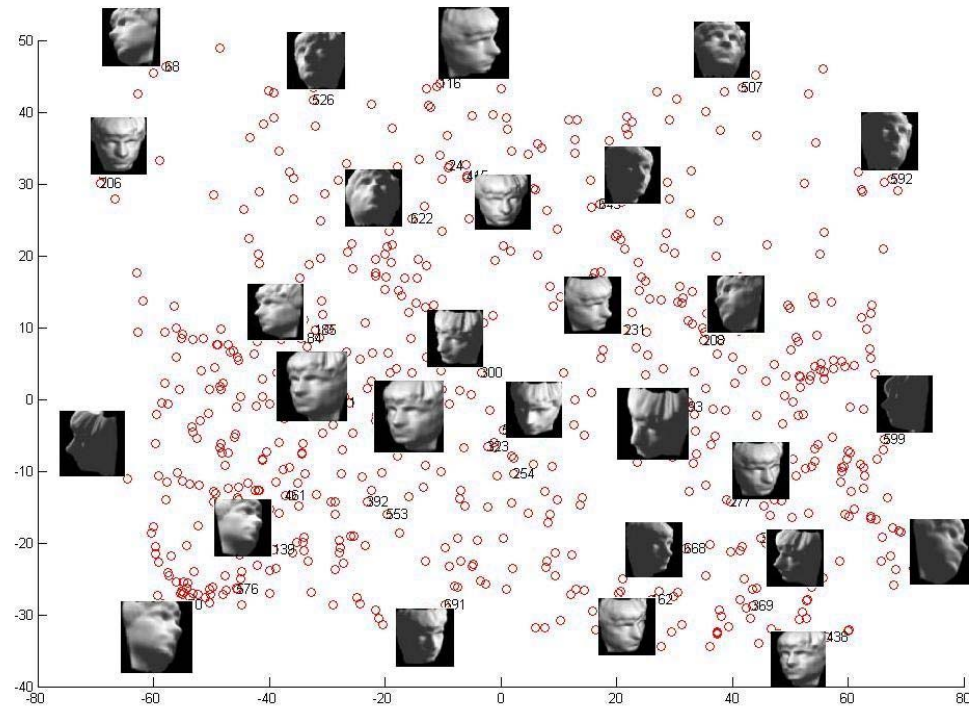


Isomap: More Results



- Same inputs, but this time with $d=3$

698 images of 64x64 K=7



BoostMap:

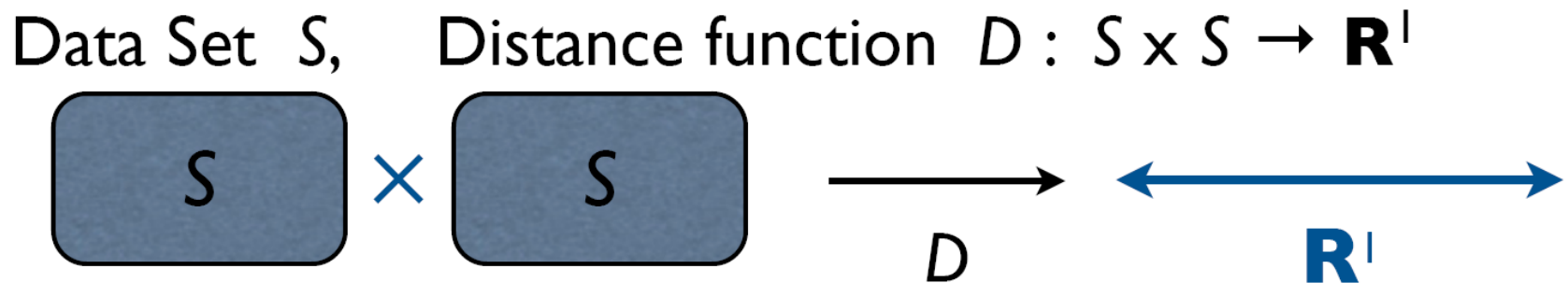
A different perspective of embedding



- Goal – Significantly reduce retrieval time in image database systems.
- Embedding is formulated as a machine learning task.
- AdaBoost is used to combine many simple 1D embeddings into a d-dimensional embedding.
- Obtain ranking of all DB objects in order of similarity to a query object.



BoostMap: main idea



Main Points:

D may be expensive to compute.

D may not satisfy triangle inequality.

BoostMap: Problem Definition



- Embeddings are seen as classifiers.
- Estimate for a, b, c if a is closer to b or c .
- X – set of objects
- D_X – distance measure.

$$P_X(q, x_1, x_2) = \begin{cases} 1 & \text{if } D_X(q, x_1) < D_X(q, x_2) \\ 0 & \text{if } D_X(q, x_1) = D_X(q, x_2) \\ -1 & \text{if } D_X(q, x_1) > D_X(q, x_2) \end{cases}$$

- Find an Embedding $F: X \rightarrow \mathbb{R}^d$ and a measure $D_{\mathbb{R}^d}$ that is used for evaluating any triplet.

$$\tilde{F}(q, x_1, x_2) = D_{\mathbb{R}^d}(F(q), F(x_2)) - D_{\mathbb{R}^d}(F(q), F(x_1))$$

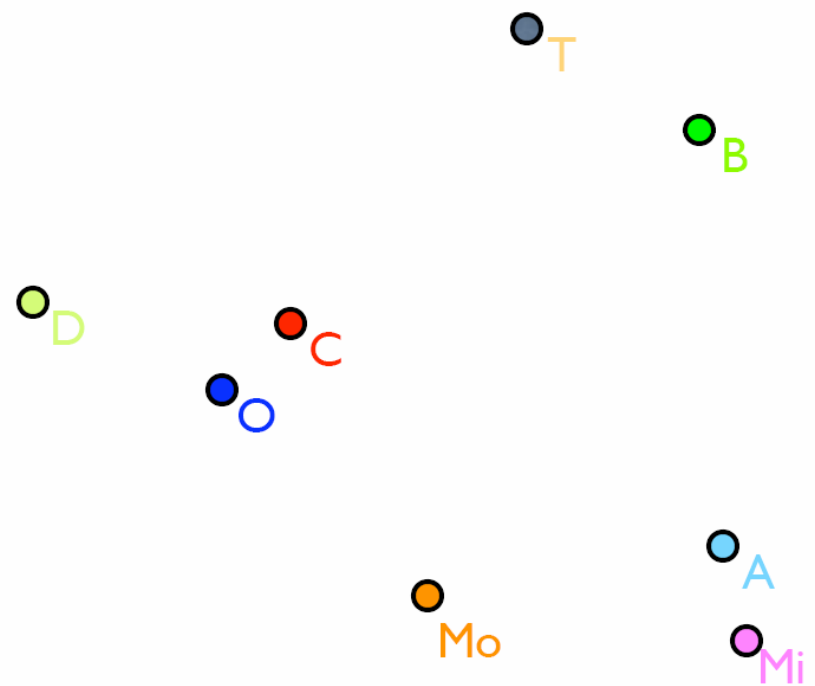


- Data Set S , Distance function $D : S \times S \rightarrow \mathbf{R}^+$
- Proximity function $P(r; x, y) = \text{sgn}(D(y, r) - D(x, r))$

Example.

Cities of North America

Some proximity function values
for “Cities of North America”





BoostMap - Outputs

- The output is a classifier: $H = \sum_{j=1}^d \alpha_j \tilde{F}_j$
- The final output is an embedding $F : X \rightarrow \mathbb{R}^d$
And a distance measure $D : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$

$$F(x) = (F_1(x), \dots, F_d(x))$$

$$D_{\mathbb{R}^d}((u_1, \dots, u_d), (v_1, \dots, v_d)) = \sum_{j=1}^d (\alpha_j |u_j - v_j|)$$



BoostMap - Results

Hand shapes
used in the
training set



Orientations
used in the
training set



Retrieval results



original

The
query

Correct
match

Summary:

Nonlinear Dimensionality Reduction



- **Isomap** - Use the geodesic manifold distances between all pairs.
 - sees more than just the Euclidean structure.
 - polynomial time procedure.
- **LLE** - Recovers global nonlinear structure from locally linear fits.
 - no need no estimate pair-wise distances.
 - optimization do not involve local minima.
- **BoostMap** - looks at embeddings as classifiers, uses AdaBoost.
 - main usage: similarity retrieval from database.
 - main advantage: trained offline, applicable online.
- Manifold learning ...



流形学习法

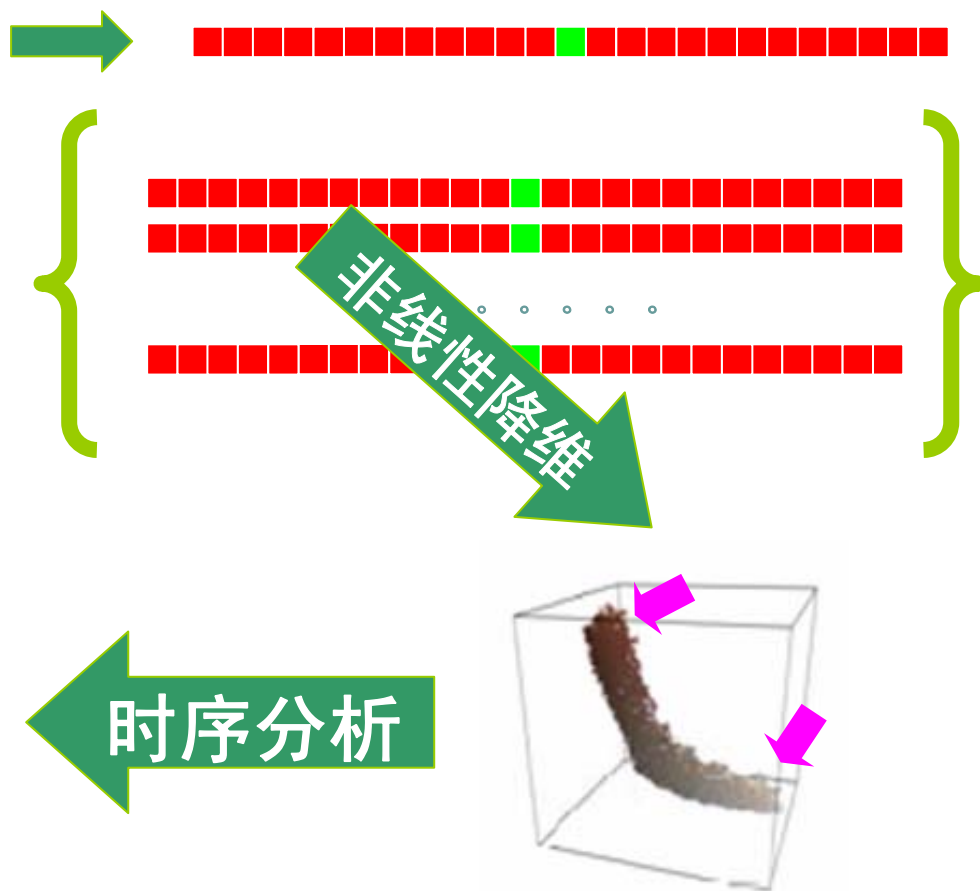
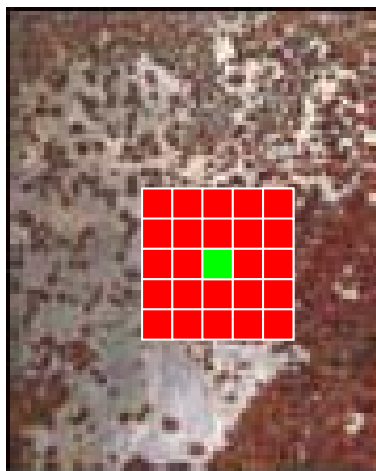
- 关键问题
 - 每个像素对应时间/年龄参数
 - 风化程度 (Weathering Degree)
 - 每个风化程度对应的纹理值

难点：单幅图像 -> 时序关系



- 流形学习 (**Manifold Learning**)
 - {高维数据} → 结构信息
非线性降维

流形学习法





流形学习法—合成结果



- 曲面上的合成方法等
- 存储、绘制等等

流形学习法—合成结果

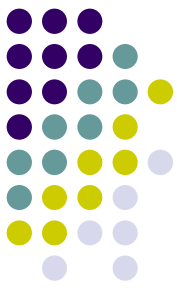


原图

做新

做旧





Homework

- Algorithm implementations
 - Eigenface
 - ISOMAP
- Read the ISOMAP, LLE and BoostMap papers.
- Keep on thinking:
 - How to use dimension reduction results